**AWS NOTES**

**New AWS Terms that we get**

**=========================**

AWS, AWS Console, AWS EC2 instance, AWS CloudFormation, AMI (Amazon Machine Image), IAM (Identity Access Management), Key Pair (Authentication), VPC, Subnet, IP-Address (IP-V4, IP-V6), Firewall (Security Group), Port, KeepAlive (Heartbeat), JSON, YAML, Visual Studio Code (VSCode), AWS CLI (a command line interface to AWS Console), Staggering (Mainly used to store files in S3 by breaking the big file into smaller sized files), URI (Uniform Resource Indicator), URL (Uniform Resource Locator), Intelligent-Tiering, CloudTrail, CRUD (Create, Read/Retrieve, Update, Delete), JSON (JavaScript Object Notation - key:value pair)

***Key pair*** - Some authentication mechanism to enable login to the virtual system/server/machine (AWS EC2 instance), from outside - that is we are going to access a machine somewhere in the internet (cloud) in some region of AWS from outside the environment - therefore we need some logic to handshake and get inside that machine. This is called connection.

**Steps to generate the key pair**

**-----------------------------------------**

1. create key pair with RSA type and with .pem, which will be downloaded to your local windows m/c - done

2. This .pem file will be converted to .ppk file for Putty to connect to the AWS instance with the same key pair encryption - done

3. Connect Putty to the instance - how do we do that -

\* Opened Putty Configuration by invoking putty

\* created a session by the name -> dsa30m

\* Host -> ip-address

\* port -> 22

\* window -> Behaviour -> Window title -> give a title to the putty window

\* Connection -> Data -> Auto-login Username -> ubuntu

\* SSH -> Auth -> Credentials -> browse -> select the .ppk file

\* Session -> Saved Session -> give session name -> Save

\* OPEN

4. Want to reconnect

\* Invoke Putty

\* Select the session from the list

\* Load -> Open

**Note:**

\* Follow the document to download Putty and install Putty into your windows system

\* *PuttyGen* -> use this utility to convert .pem file to .ppk

-> Load

-> File Name: \*.pem -> press enter -> select the .pem file -> Open -> Save Private Key

-> save the file to the same folder where you have saved the .pem file

**Security Group (F/W)**

**----------------------------**

1. We create a security group for our instance and let at least an SSH connection (22) to connect from anywhere and by any ip-address

SSH - Secured Shell connection -> SSH -> port 22 -> anywhere 0.0.0.0/0

2. We can add further rules -> HTTP -> port 80 -> anywhere 0.0.0.0/0 HTTPS -> 443 -> anywhere 0.0.0.0/0

**CLOUD Formation (Automatically [stack] create an instance using Cloud Formation services of AWS cloud)**

**--------------------------------------------------------------------------------------------------------------------------------------------**

1. Use a template - download from the template from Teams -> "Ubuntu-linux-lab-setup2.yml"

2. Cloud Formation -> Create a stack -> With new resources (standard) -> Template is ready -> Upload a Template file -> Choose file -> Next -> Stack Name -> Next -> Next -> Submit

3. Check the instances and verify the instance has been created correctly with the attached Key-Pair and Security-Group

4. Connect using Putty

5. Connect using GitBash

$> ssh -i "c:/aws/aws-key-files/dsa30m.pem" ubuntu@<public-ip-address>

**USERDATA Demo**

**-----------------------**

**EC2: userdata**

============

#!/bin/bash

sudo apt update -y

sudo apt install apache2 -y

sudo systemctl start apache2.service

sudo systemctl enable apache2.service

touch /tmp/myfile.txt

useradd sophie

useradd derry

useradd moses

mkdir /tmp/kalkey

mkdir /tmp/hello

chown sophie:sophie /tmp/hello

echo "<h2>Hello Everyone this is a BACKUP SERVER OF KALKEY.COM $(hostname -f) </h2>" > /var/www/html/index.html

**IAM (Identity Access Management)**

**-----------------------------------------------**

\* Create a IAM user (use the IAM service of AWS)

\* Create the Access Keys

\* Access Key ID (AKIAWT3Y7LSYFFO3JXMP) (ZWbB4BxL4gEOPK29KZ3KdaNgnplgqCp3P98e/DmF)

\* Secret Access Key (this is generated only once and cannot be retrieved in future, so pls store it somewhere)

\* Create the Role policy/Permissions - Policy name = AdministratorAccess

**S3 (Simple Storage Service)**

**S3 Storage (Understanding & Creating Bucket, Versioning, Static Website )**

**-----------------------------------------------------------------------------------------------------**

\* Cloud Storage - uses, pay as u use, ease of storage, ease of access

\* Types of storage

- S3 - Simple storage as discussed above

- EBS - Elastic Block Storage - SSD hard drives that get attached to our system/instance

- EFS - Elastic File Storage - unlike EBS which is attached to an instance EFS is shared accross the instances

- Glacier - Archiving soln over AWS cloud - to dump the data for as low cost as possible then Glacier is the soln

- Storage Gateway - Move the data easily from local to Cloud or have a synchronization between the local and the Cloud for internet users to access the same data as is used locally

- Snowball - Data import and export system

- Snowball edge - where data in form of hardware is shipped our premises where we copy the data and ship it back to Amazon and Amazon will copy the data and

ship/transport/transfer/export them further to whichever destination we wanted

- Snowmobile - if the data is huge than we can call a Snowmobile which is Data Center on a truck, which carries the data back to Amazon to be exported to any other datacenter or destination

which we required

\* S3 is an object storage -

\* that is we can store data from internet, we can retrieve from internet, but we cannot install anything in S3, unlike the block storage. S3 file size can be as big as 5 TB and there can be 1000s of

such files stored in S3.

\* S3 can be used as a code repositories by developers, share artifact with encryption

\* Durability and availabilty is 11 9s (99.999999999) and 99.99% respectively

\* Low Cost

\* Scalability

\* Security (Bucket policies and encryption)

\* Flexibility

\* Bucket & Object -> Object is the data + Metadata (what is type of data, author or some non-functional data) on the S3 and Bucket is the container or the Folder where the data is stored/accessed

- Demo on S3 creation, upload, access

- Bucket policies - who accesses what is like IAM policy

{

"Id": "Policy1683024249764",

"Version": "2012-10-17",

"Statement": [

{

"Sid": "Stmt1683024244658",

"Action": [

"s3:GetObject"

],

"Effect": "Allow",

"Resource": "arn:aws:s3:::dsa30m-2ndmay2023/\*",

"Principal": "\*"

}

]

}

- Demo Version Control on S3 - roll forward/roll back

- Storage Classes -

- S3 standard for frequent data access

- S3 standard for long lived and infrequent data access

- Amazon Glacier for an archival solution in cloud

- One Zone-1A availability zone data storage for in frequent data access needs to avoid multiple az's storage to avoid charges

- Amazon S3 Standard reduced redundancy storage - library books which is non critical and replaceable if lost

- Lifecycle management

- Moving files not ferquently used from S3 standard storage to S3 infrequent access storage and further to S3 Glacier, which saves cost

- This is called Transition actions

- THe other one is Expiration actions (Telecom example of cdrs)

- Demo on lifecycle management

- CloudFront - Transfer Acceleration feature of S3

- CloudFront is a content delivery network (CDN) service that securely transfers data to your preferred destination with a high transfer speed

**S3 Static Website Hosting**

**----------------------------------**

\* Create S3 bucket

\* upload web page and image to S3 bucket

\* specify bucket for static website

\* unblock public access for policies

\* add bucket policy to allow reading buckets (get objects)

Index.html

---------------

<html>

<head>

<title> S3 Hosted Website</title>

</head>

<body>

<h1> My S3 Hosted Website </h1>

<p> This website is hosted in AWS S3 </p>

<img src=".../images/hello\_world.jpg" />

</body>

</html>

Error.html

-------------

<h1><font color="green"> Oops! There's an error! </font></h1>

**TASKS:**

* Create a bucket
* Upload some file
* Try open it
* Try to open it from URL (public)
* From permission section provide public access enable ACL
* Try to open it now
* Enable version controlling from properties tab of bucket
* Upload same files few times and check if show version option is showing up or not
* Check the version ids data/time stamp

**EC2 Load Balancer**

**-------------------------**

Define/Create Classic LB

Create a separate SG for load balancer

ELB prerequisite:

1. Launch 2 ec2 instances. (using cloudformation and labsetup.yml file).

2. Give them the name web1 and web2.

3. Install httpd and start the service.

4. Go to the /var/www/html and create index.html and write something on it. (not identical).

5. From the security group for both the instance allow port 80 from anywhere.

6. Go to the browser and check if pages are being loaded from both the instances or not (using the IP address).

**AutoScaling Practice:**

**----------------------------**

1. Click to the ec2 dashboard.

2. Click autoscaling group (in left hand side list)

3. Create autoscaling group.

4. Create launch template -> give a name -> selcet AMI (amazon linux 2)

--> instance type (t2 micro the free one) --> select a key pair or create one.

--> select security group. \*\*keep rest of the options as it is.

5. Click on create launch template. (not required).

6. Now go to the autoscaling group.

7. Create autoscaling group. select the template and next.

8. Adhere to the launch template.

9. VPC (default). Subnets select all of them (the AZs).

I have a loadbalancer called dsa18ELB: u include the ec2 instances.

**Amazon RDS - Relational Database Services - CRUD [ Create, Read/Retrieve, Update, Delete ]**

**-----------------------------------------------------------**

**Demo to create the RDS database instance on AWS (MariabDB)**

**Demo to connect to the RDS database with an EC2 instance having Database Client (Mariadb)**

**Install MariaDB on Ubuntu Server/Instance**

$> sudo apt update

$> sudo apt install mariadb-server -y

----- Configuring MariaDB -----

$> sudo mysql\_secure\_installation

----- Follow the prompts and respond as given below

<enter>, n, n, n, y, y, y

----- Creating an Administrative User that Employs Password Authentication -----

$> sudo mariadb

MariaDB [(none)]> GRANT ALL ON \*.\* TO 'admin'@'localhost' IDENTIFIED BY 'password' WITH GRANT OPTION;

MariaDB [(none)]> FLUSH PRIVILEGES;

MariaDB [(none)]> exit

**Testing MariaDB**

$> sudo systemctl status mariadb

$> sudo mysqladmin version

**Check MySQL client version embedded withing Mariadb server installed**

$> mysql --version

**From AWS RDS console for RDS database created go to tab "Connectivity & Security"**

Database -> mydb

endpoint -> mydb.claemgzejhja.ap-southeast-1.rds.amazonaws.com

port -> 3306

----- Go to "Configuration" tab to get the Master User Name -----

master user name -> admin

password -> password

**AWS Cli - using AWS cli we can get the above information as well**

$> aws rds describe-db-instances \

--filters "Name=engine,Values=mariadb" \

--query "\*[].[DBInstanceIdentifier,Endpoint.Address,Endpoint.Port,MasterUsername]"

**Connecting from the MySQL command-line client (unencrypted)**

$> mysql -h <endpoint> -P 3306 -u <mymasteruser> -p [ syntax]

$> mysql -h mydb.claemgzejhja.ap-southeast-1.rds.amazonaws.com -P 3306 -u admin -p

password: password

----- you should get connected: -------

MariaDB [(none)]>

MariaDB [(none)]> CTRL + l (to clear the screen)

MariaDB [(none)]> show databases; (to list the databases)

MariaDB [(none)]> create database awsclass; ( to create a database by name "awsclass" )

MariaDB [(none)]> use awsclass; ( to select a database )

MariaDB [(awsclass)]> show tables; ( to list all the tables in the database "awsclass" )

MariaDB [(awsclass)]> drop database awsclass; ( to drop/delete a database )

MariaDB [(none)]> create database awsclass;

MariaDB [(none)]> use awsclass;

**----- To create a table "students" in database "awsclass" -----**

MariaDB [(awsclass)]> create table students (

id int not null,

name varchar (20) not null,

age int not null,

course char (10),

salary decimal (15,2),

primary key (id)

);

MariaDB [(awsclass)]> show tables;

MariaDB [(awsclass)]> describe students; ( to check the table details structure )

**----- To add DATA into the table "students" -----**

MariaDB [(awsclass)]> insert into students (id,name,age,course,salary) values (1,'Sophie',25,'linux',2000.00);

MariaDB [(awsclass)]> insert into students (id,name,age,course,salary) values (2,'Moses',28,'AWS',3000.00);

MariaDB [(awsclass)]> insert into students (id,name,age,course,salary) values (3,'Derry',30,'Docker',3500.00);

MariaDB [(awsclass)]> insert into students (id,name,age,course,salary) values (4,'Debasis',49,'linux',1000.00);

MariaDB [(awsclass)]> select \* from students; ( to query all the rows from students table )

**----- To query a specific details: -----**

MariaDB [(awsclass)]> select \* from students where name = 'Derry';

MariaDB [(awsclass)]> select \* from students where salary > 2000;

MariaDB [(awsclass)]> select name,age from students where salary <=2000;

MariaDB [(awsclass)]> select \* from students where age >21 and salary >2000;

MariaDB [(awsclass)]> select \* from students where age >30 and salary >100;

**----- To update table: -----**

MariaDB [(awsclass)]> update students set course='ansible' where id=2;

**----- To check permissions/privileges: -----**

MariaDB [(awsclass)]> show grants;

**----- To get help: -----**

MariaDB [(awsclass)]> help

MariaDB [(awsclass)]> help contents

MariaDB [(awsclass)]> help account management

MariaDB [(awsclass)]> help create user

**----- To create user: -----**

MariaDB [(awsclass)]> create user 'debasis'@'localhost' identified by 'password';

**----- To give privilege to select for the user: -----**

MariaDB [(awsclass)]> grant select,update on students to debasis@localhost;

==================================================================================================================================